The Application Layer: SMTP, FTP

CS 352, Lecture 5

http://www.cs.rutgers.edu/~sn624/352-S19

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Recap: Application-layer protocols

- DNS: lookup a (machine-readable) address using a (human-readable) name?
  - Used by many applications as a building block
  - Recursive lookups for scale and administrative convenience

- HTTP: Transfer content over the Internet with portability
  - Very familiar to users of the Internet
  - Many, many, applications use HTTP

- Today: SMTP (email) and FTP (file transfer)

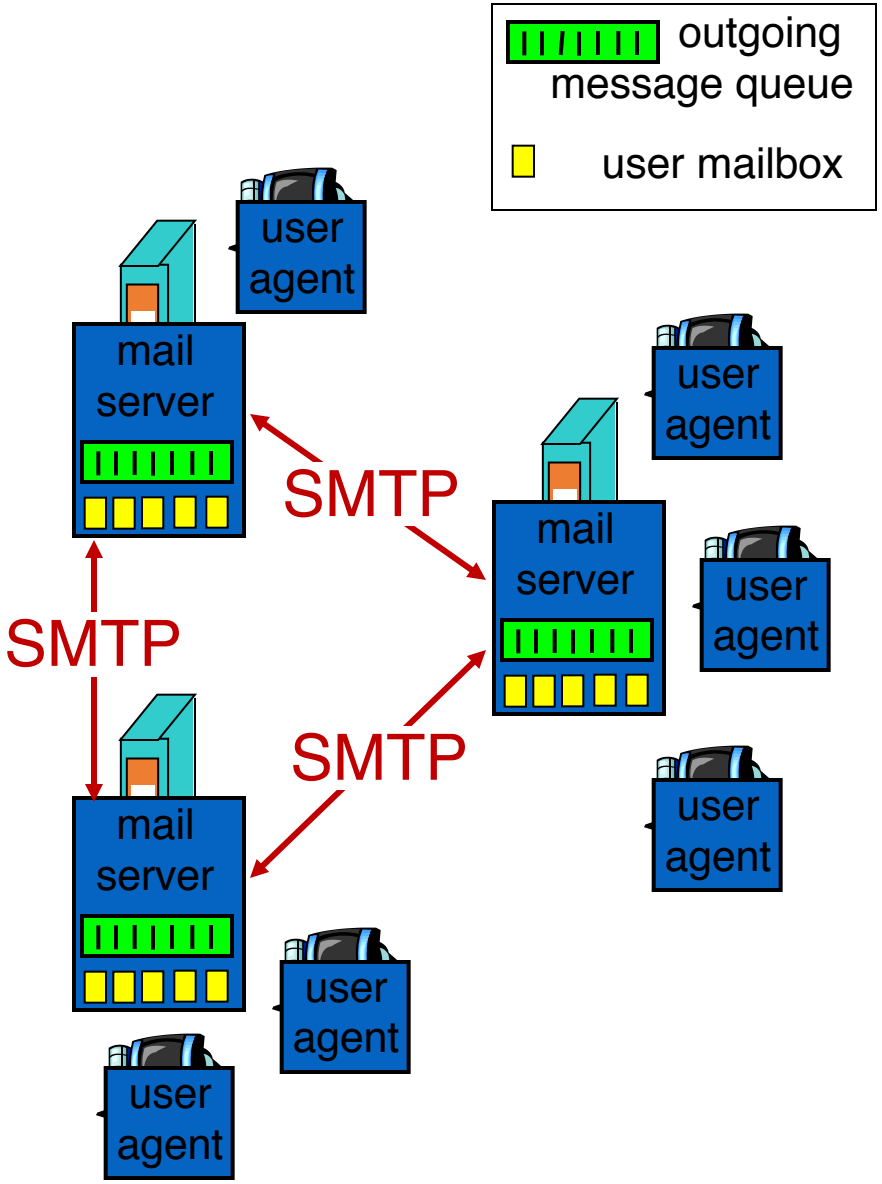
SMTP

Simple Mail Transfer Protocol

Electronic Mail

Three major components:

1. User agents
   - a.k.a. “mail reader”
   - e.g., Applemail, Outlook
   - Web-based user agents (ex: gmail)



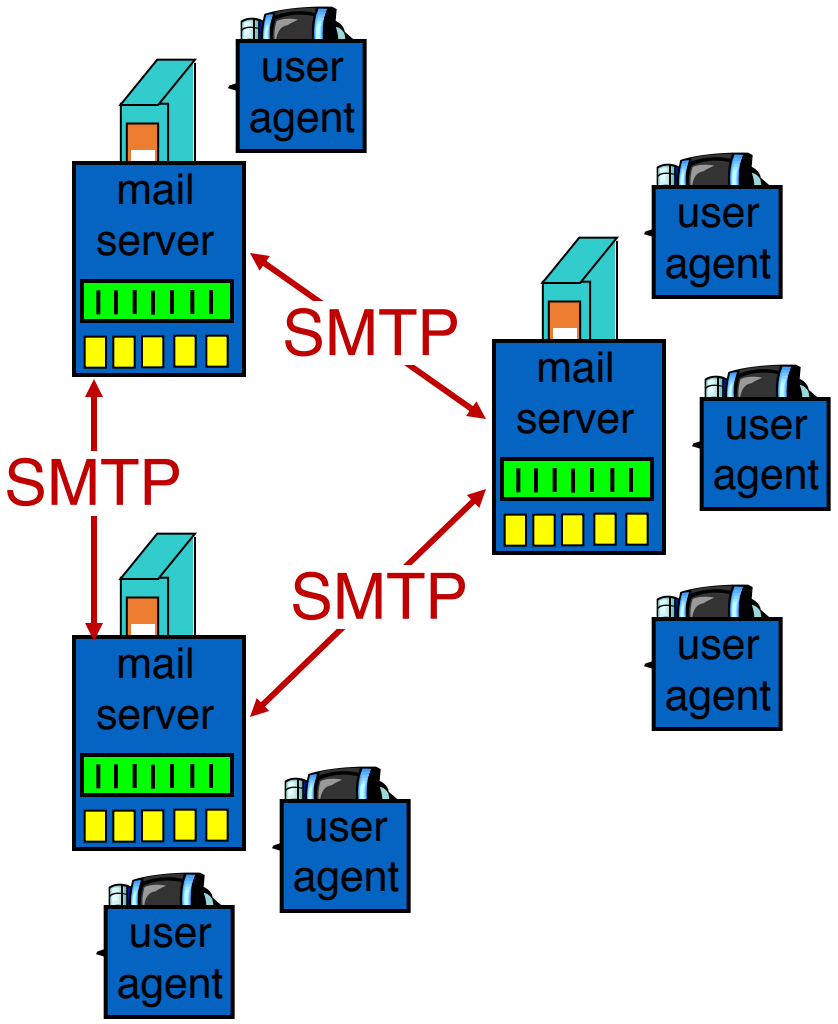
Electronic Mail: Mail servers

2. Mail Servers

- Mailbox contains incoming messages for user
- Message queue of outgoing (to be sent) mail messages
- Sender mail server makes connection to Receiver mail server
  - IP address, port 25

3. SMTP protocol

- Used to send messages
- Client: sending user agent or sending mail server
- server: receiving mail server



Scenario: Alice sends message to Bob

1) Alice uses UA to compose message and "to" `bob@someschool.edu`

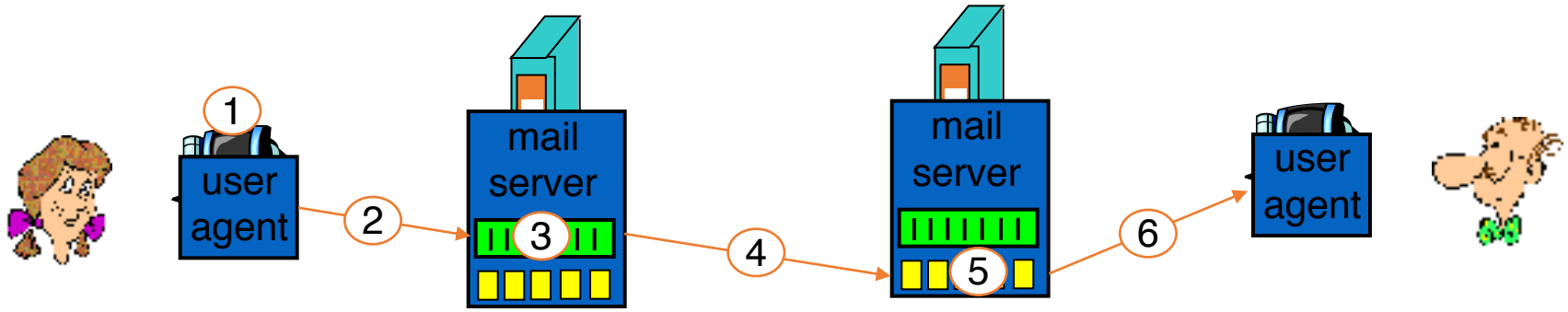
2) Alice's UA sends message to her mail server; message placed in message queue

3) Client side of SMTP opens TCP connection with Bob's mail server

4) SMTP client sends Alice's message over the TCP connection

5) Bob's mail server places the message in Bob's mailbox

6) Bob invokes his user agent to read message



Sample SMTP interaction

220 hill.com SMTP service ready

HELO town.com

250 hill.com Hello town.com, pleased to meet you

MAIL FROM: <jack@town.com>

250 <jack@town.com>… Sender ok

RCPT TO: <jill@hill.com>

250 <jill@hill.com>… Recipient ok

DATA

354 Enter mail, end with "." on a line by itself

Jill, I'm not feeling up to hiking today. Will you please fetch me a pail of water?

.

250 message accepted

QUIT

221 hill.com closing connection

MAIL command response codes

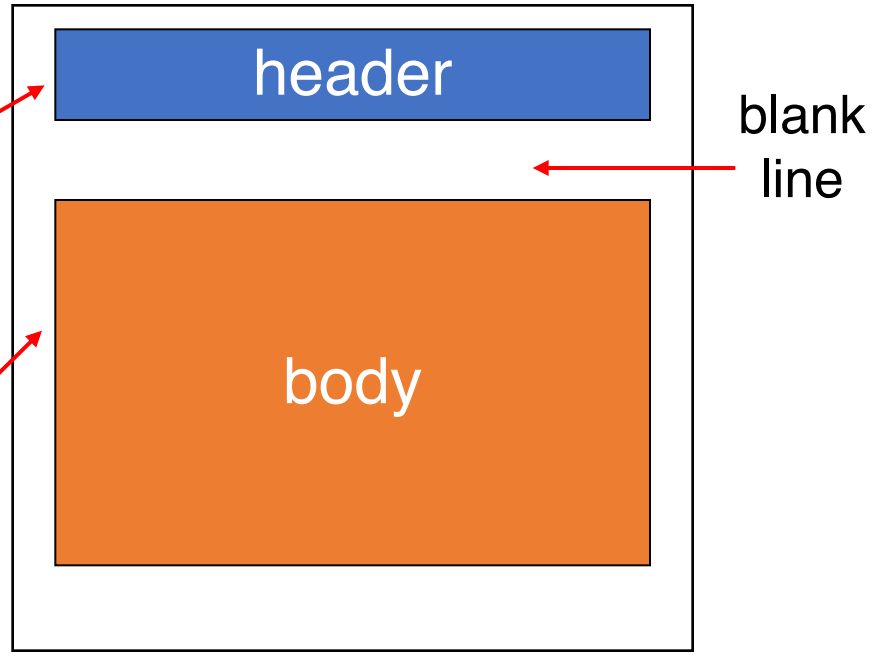
Table 23.2 *Responses*

Code	*Description*
Positive Completion Reply	
211	System status or help reply
214	Help message
220	Service ready
221	Service closing transmission channel
250	Request command completed
251	User not local; the message will be forwarded
Positive Intermediate Reply	
354	Start mail input
Transient Negative Completion Reply	
421	Service not available
450	Mailbox not available
451	Command aborted: local error
452	Command aborted; insufficient storage
Permanent Negative Completion Reply	
500	Syntax error; unrecognized command
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command temporarily not implemented
550	Command is not executed; mailbox unavailable
551	User not local
552	Requested action aborted; exceeded storage location
553	Requested action not taken; mailbox name not allowed
554	Transaction failed

Mail message (stored on server) format

SMTP: protocol for exchanging email msgs

RFC 822: standard for text message format:



- header lines, e.g.,
  - To:
  - From:
  - Subject:

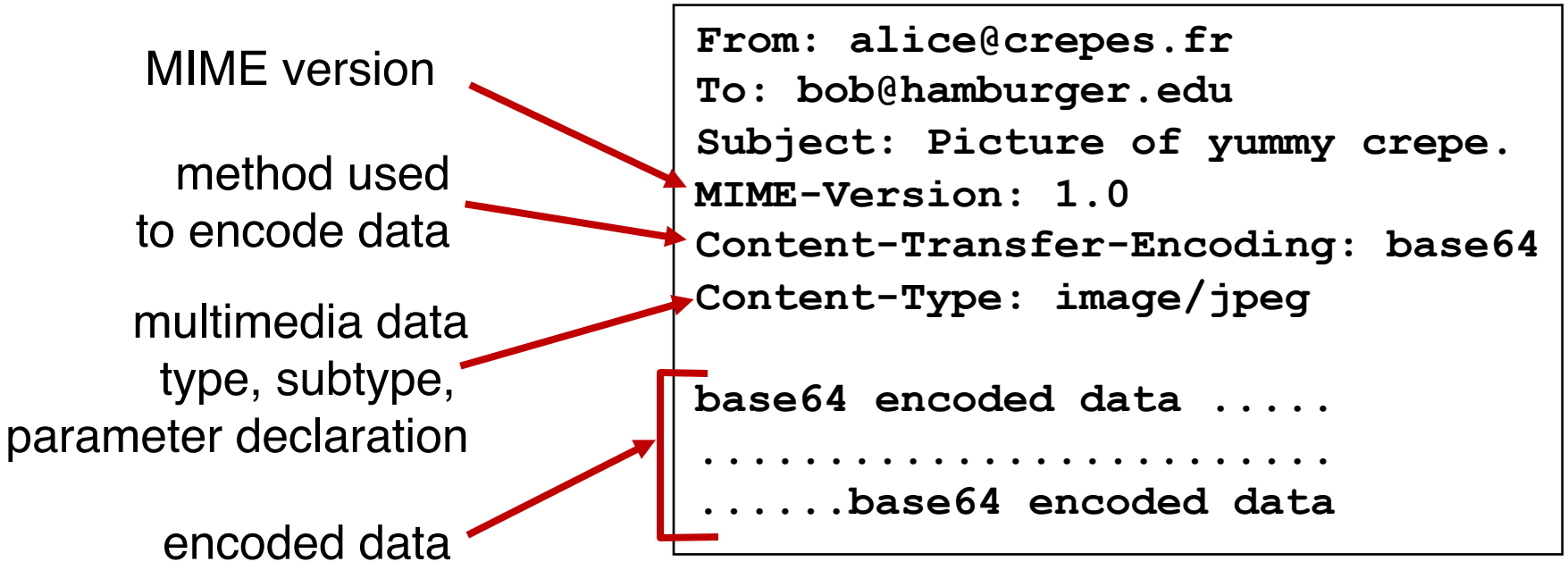
  different from SMTP commands!

  (these would still be under “DATA”)

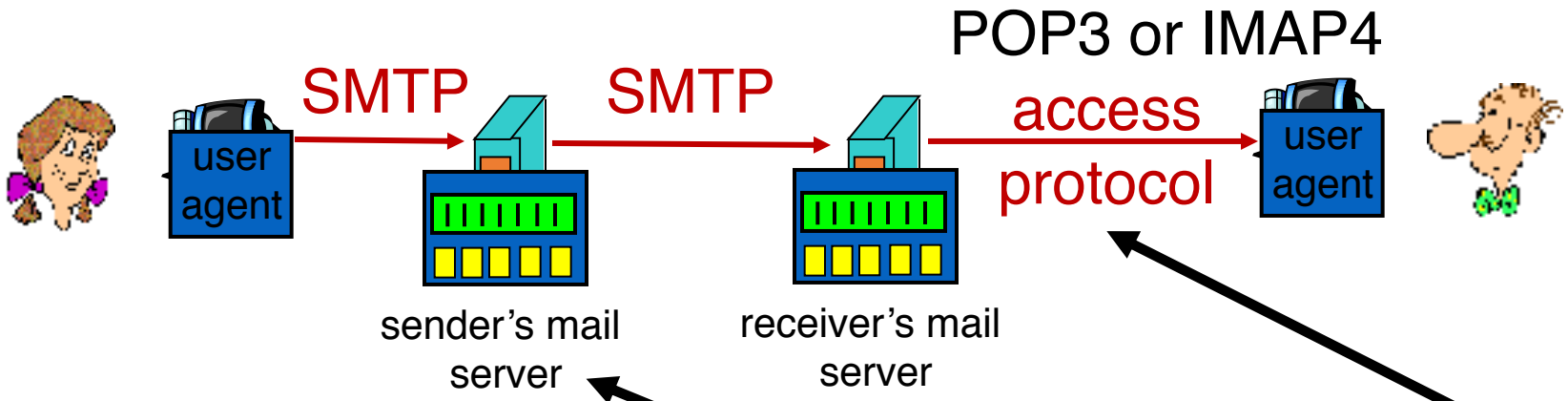
- body
  - the “message”, ASCII characters only

Message format: multimedia extensions

- MIME: multimedia mail extension, RFC 2045, 2056
- additional lines in msg header declare MIME content type



Mail access protocols



- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]
    - Client connects to POP3 server on TCP port 110
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - Client connects to TCP port 143
  - HTTP: gmail, Yahoo! Mail, etc.

Why not use SMTP here?

Why do we need a sender side mail server?

POP vs IMAP

- POP3
- Stateless server
- UA-heavy processing
- UA retrieves email from server, then typically deleted from server
- Latest changes are at the UA
- Simple protocol (list, retr, del within a POP session)

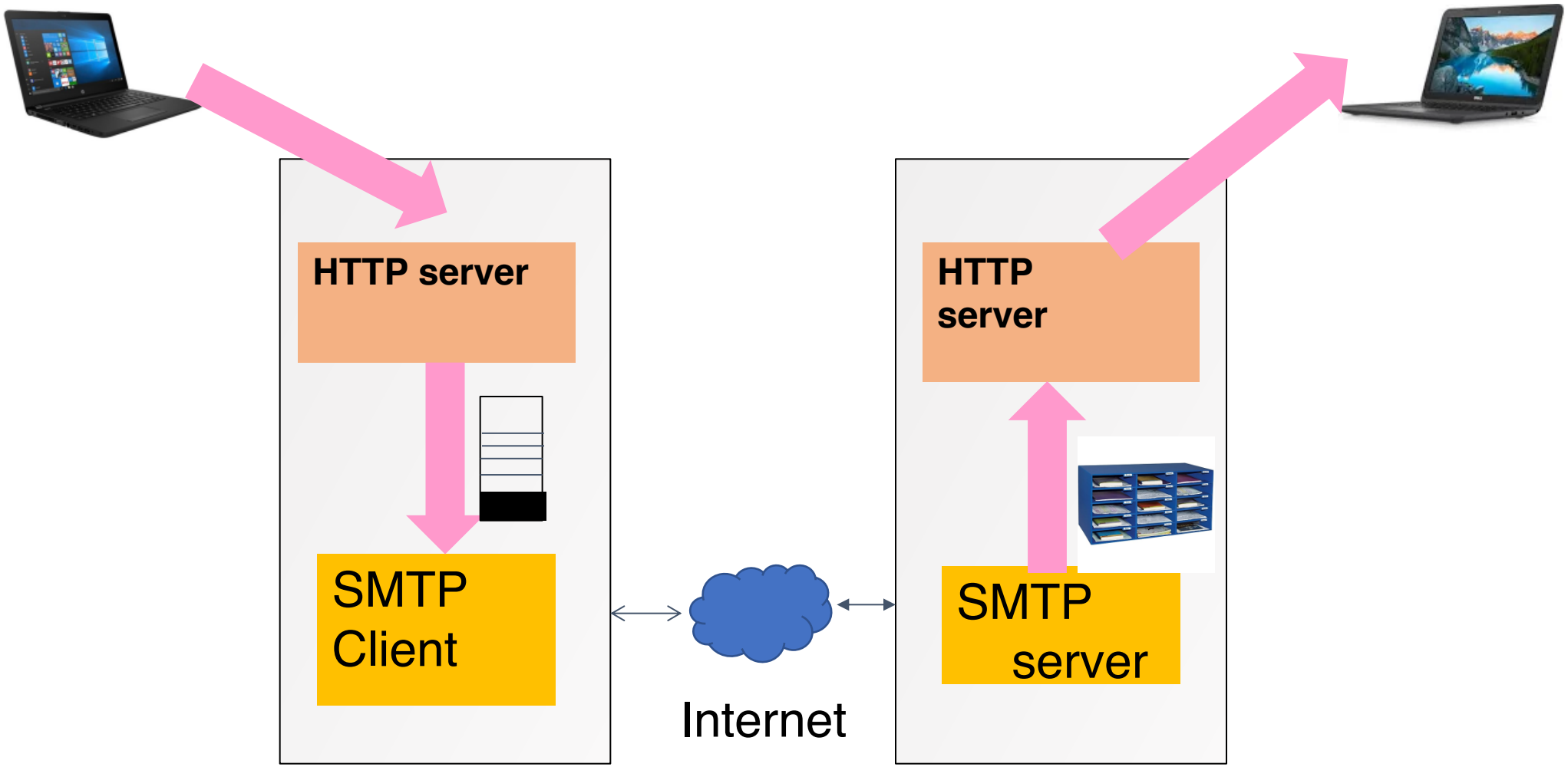
- IMAP4
- Stateful server
- UA and server processing
- Server sees folders, etc. which are visible to UAs
- Changes visible at the server
- Complex protocol

What about web-based email?

- Connect to mail servers via web browser
  - Ex: gmail, outlook, etc.

- Browsers speak HTTP
- Email servers speak SMTP
- Need a bridge to retrieve email using HTTP!

Web based email



Comparing SMTP with HTTP

- HTTP: pull
- SMTP: push

- both have ASCII command/response interaction, status codes

- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg

- HTTP: can put non-ASCII data directly in response
- SMTP: need ASCII-based encoding!

Try an SMTP interaction!

```
ngsrinivas@ubuntu18-vbox:~$ nslookup
> set type=MX
> rutgers.edu
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
rutgers.edu     mail exchanger = 10 mx.rutgers.edu.

Authoritative answers can be found from:
>
ngsrinivas@ubuntu18-vbox:~$ telnet mx.rutgers.edu 25
Trying 128.6.68.142...
Connected to mx.rutgers.edu.
Escape character is '^]'.
220 mx.rutgers.edu ESMTP
HELO cs.rutgers.edu
250 annwn11.rutgers.edu
MAIL FROM: <sn624@cs.rutgers.edu>
250 2.1.0 Ok
RCPT TO: <srinivas.narayana@rutgers.edu>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Hello, world!
Goodbye, cruel world.
.
250 2.0.0 Ok: queued as 2B2E5460035
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
ngsrinivas@ubuntu18-vbox:~$ ^C
```

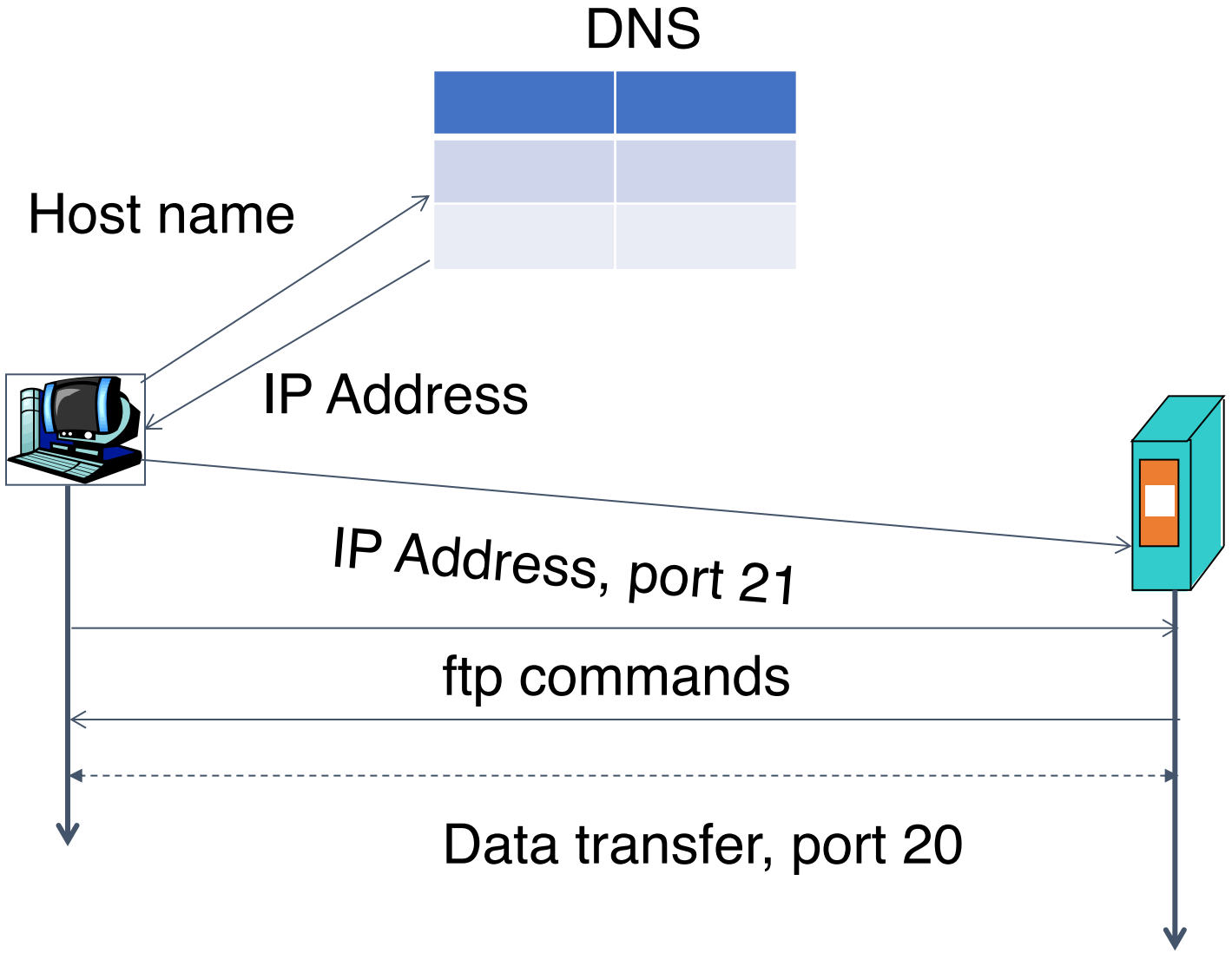
```
[flow:~]$  telnet mx.rutgers.edu 25
Trying 128.6.68.142...
Connected to mx.rutgers.edu.
Escape character is '^]'.
220 mx.rutgers.edu ESMTP
HELO cs.rutgers.edu
250 annwn12.rutgers.edu
MAIL FROM: <sn624@cs.rutgers.edu>
250 2.1.0 Ok
RCPT TO: <srinivas.narayana@rutgers.edu>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
From: sn624@cs.rutgers.edu
To: srinivas.narayana@rutgers.edu
Subject: A test message

Hello. Bleh bleh bleh.
.
250 2.0.0 Ok: queued as 904AA634015
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

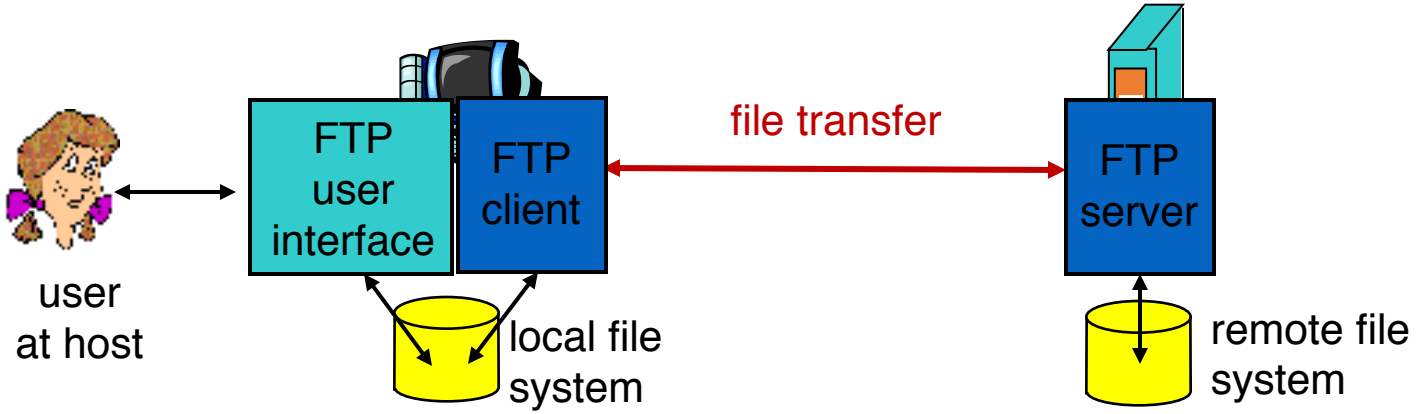
FTP

File Transfer Protocol

Client server connection



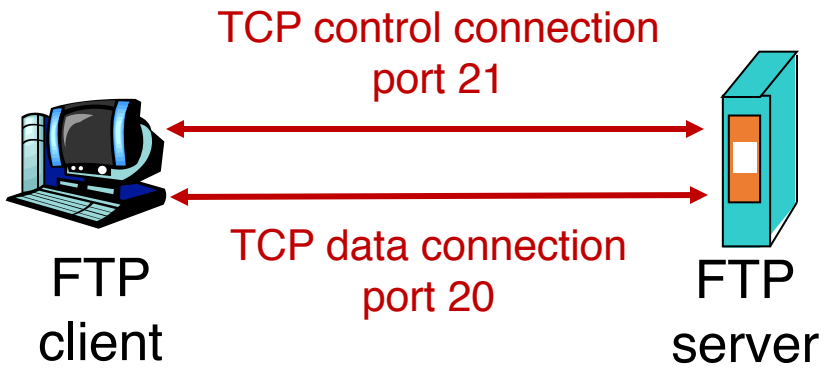
FTP: the file transfer protocol



- transfer file to/from remote host
- client/server model
  - *client:* side that initiates transfer (either to/from remote)
  - *server:* remote host
- ftp: RFC 959
- ftp server: port 21, port 20 (data connection)

FTP: separate control & data connections

- “out of band” control
  - Control connection:
    - Authorization
    - Directory browse
    - Commands
  - Data connection
    - Transfer files
- FTP server maintains “state”: current directory, earlier authentication



FTP commands, responses

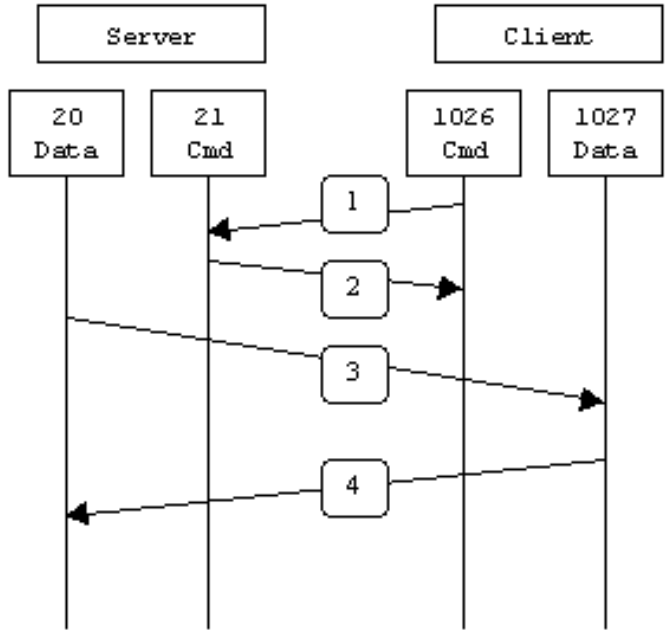
Sample commands

- sent as ASCII text over control channel
- **`USER`** ***`username`***
- **`PASS`** ***`password`***
- **`LIST`** return list of file in current directory
- **`RETR filename`** retrieves (gets) file
- **`STOR filename`** stores (puts) file onto remote host

Sample return codes

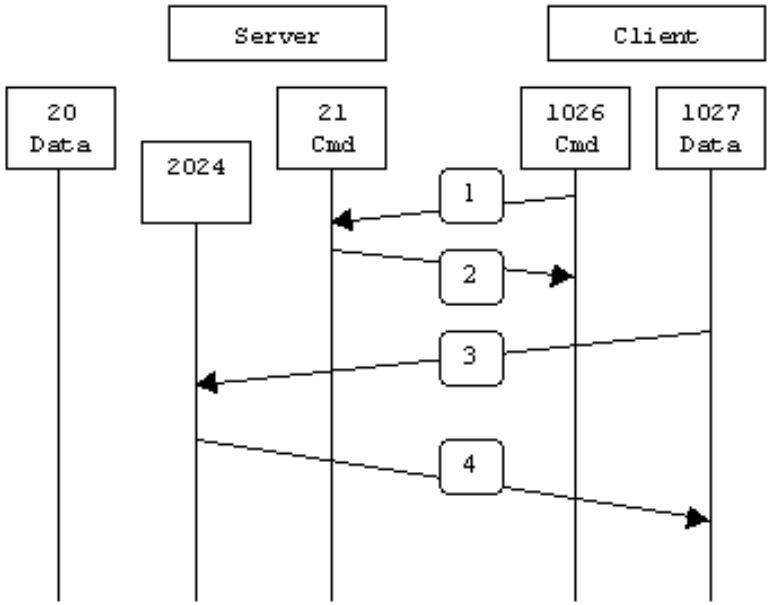
- status code and phrase (as in HTTP)
- **`331 Username OK, password required`**
- **`125 data connection already open; transfer starting`**
- **`425 Can't open data connection`**
- **`452 Error writing file`**

FTP Active connection



- Data connection initiated from the server
- Client opens a connection from port x for sending commands to server port 21
- Server opens a connection from port 20 to send data at port x+1

FTP passive connection (always client initiated)

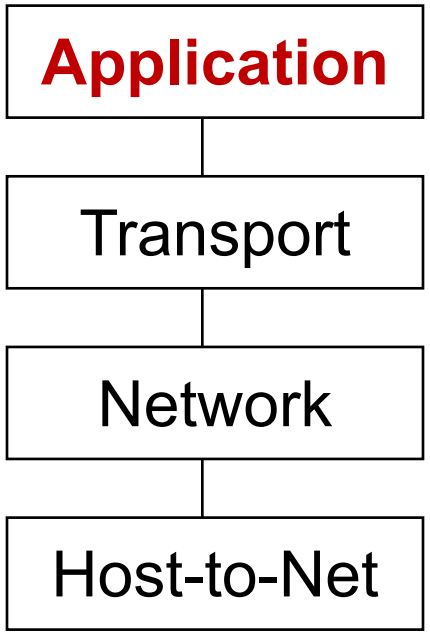


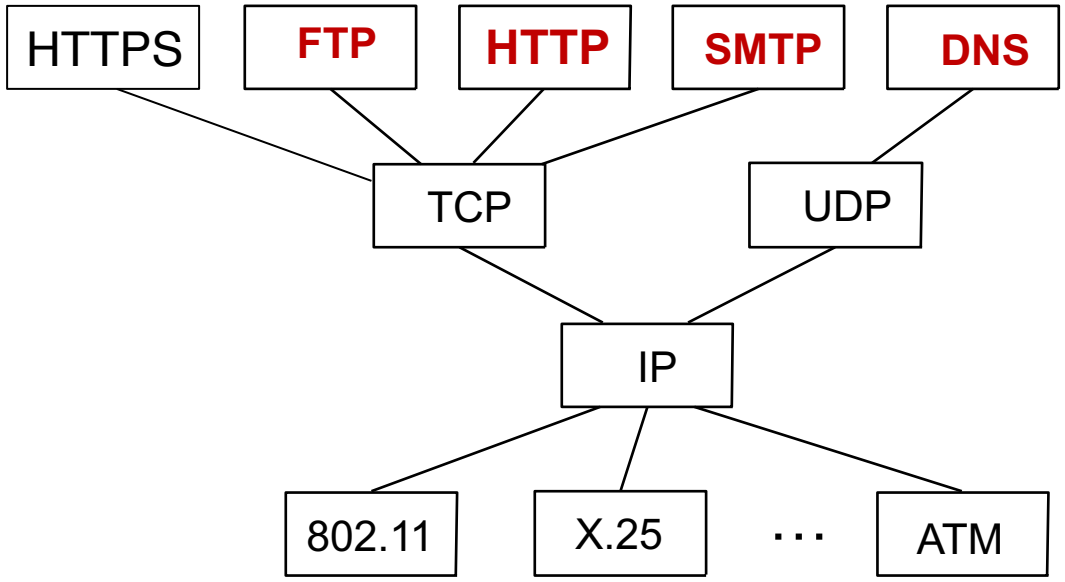
- Connections always initiated from the client
- Client opens a connection from port x for sending commands to server port 21
- Client sends a request for PASSIVE connection with PASV command
- Server replies with a new port number S_p on which it is listening
- Client opens a connection from port x+1 to server port S_p

Problems with FTP

- Sends passwords in plain ASCII text
  - Eavesdropper can recover passwords
  - Fatal flaw, turned off at a lot of sites
  - Replaced with scp, sftp instead

Recall the Internet protocol stack…





Themes from application-layer protocols

- Request/response nature of protocols
  - Headers determine the actions of all the parties of the protocol
- Separation of concerns: Examples:
  - Content rendering for users (browser, UA) separated from protocol ops (mail server)
  - Reliable mail reception: Retrieve email at an “always on” receiver mail server
  - Reliable mail sending: Send mail using a different machine rather than UA
- In-band vs. out-of-band control: SMTP+HTTP vs. FTP
  - Q: How are commands recognized as distinct from data?
- Keep it simple until you really need complexity
  - Examples: ASCII-based design; stateless servers
  - IMAP for email organization
  - Secure extensions for FTP and HTTP

Next: Transport